CRT-2001-21

Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

A FRAMEWORK FOR INFORMATION MANAGEMENT IN DYNAMIC VEHICLE DISPATCHING

by

Ilham Benyahia
Jean-Yves Potvin

May 2001

PUBLICATION
CRT-2001-21

A FRAMEWORK FOR INFORMATION MANAGEMENT IN DYNAMIC VEHICLE DISPATCHING

by

Ilham Benyahia[1]
Jean-Yves Potvin[2]

* This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

[1] Université du Québec à Hull, 101, rue St-Jean-Bosco, C.P. 1250, succursale B, Hull, Canada J8X 3X7 (benyahia@uqah.uquebec.ca)

[2] Centre de recherche sur les transports, and Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 (potvin@iro.umontreal.ca)

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2001

102-02941/

Résumé. Les nouvelles technologies en traitement de l'information et en télécommunications permettent maintenant de recevoir et gérer de grandes quantités d'informations. Ces nouveaux développements sont à la source de l'émergence de systèmes informatisés pour des applications complexes en temps réel, comme les systèmes de répartition de véhicules en transport. De tels systèmes sont difficiles à concevoir et à développer puisqu'ils doivent être en mesure de répondre adéquatement à des événements souvent imprévisibles. Afin de faciliter le développement de tels logiciels, nous proposons une architecture générique distribuée, faisant appel à des agents qui peuvent être spécialisés afin d'accomplir les tâches reliées à une application spécifique. Des extensions et spécialisations à l'architecture de base sont ainsi proposées pour réaliser la tâche de répartiteur de véhicules.

Mots clés: architecture générique, agents, gestion de l'information, temps réel, répartition de véhicules.

Abstract. Emerging technologies in computing and communications now allow the processing of large amounts of real-time information. This enables the development of computerized systems which support decision making in complex dynamic environments, like vehicle dispatching in the transportation sector. These systems are difficult to design and develop since they require the real-time processing of unexpected events. A generic framework is proposed here to facilitate the development of these complex systems. The framework is based on a distributed multi-agent architecture, where the behavior of each component can be specialized to meet the requirements of a specific application. Extensions and specializations are described to support the design of a dynamic vehicle dispatcher.

Keywords: framework, agents, information management, real-time, vehicle dispatching.

1. Introduction

Emergent technologies in computing and telecommunications bring a new dimension when addressing complex real-world applications. These applications are now typically distributed and must continuously interact with their environment to acquire information that must be processed within specified time-limit. A generic model for such applications can be represented as a management system that monitors and controls its environment. This model is made up of several distributed entities, including the managers and managed entities. When significant events occur, the managed entities inform their managers about the need to process these events to maintain or acquire a required performance level (or quality of service, depending on the application domain). Managers then take decisions to guide and control the managed entities towards that goal.

We focus here on such a management system in the transportation domain, with a particular emphasis on vehicle dispatching applications. These are found, for example, in courier services, transportation-on-demand or dial-a-ride systems, police and ambulance services, and many others. Although each application has its own peculiarities, they all basically require the assignment in real-time of new transportation requests to vehicles. When consolidation is permitted (i.e., when many different requests may be assigned to the same vehicle), it must also be decided where to insert the new request in the vehicle's planned route. Clearly, the parameters on which the dispatching decisions are based may change in a non predictable manner (e.g., the travel times may drastically change following an incident on the network). The design of a system that would allow an appropriate processing of the dispatcher's decision parameters with regard to the current dispatching situation is a difficult task which is compounded by the need to meet time constraints. A generic framework that would support these requirements for vehicle dispatching would thus be very helpful.

The management framework proposed here is defined in [1], [2]. It is characterized by a distributed architecture made up of three categories of components called agents. The managed entities are the environment components which are represented by their behavior. When significant changes occur, the managed entities interact with the manager agents (also called complex agents) which are specialized in event processing. The complex agents interact also with supervisor agents when their processing load reaches a predefined threshold. The supervisor agents have complete visibility over all manager agents

and delegate information processing tasks among those agents to maintain the system required performance level. Each complex agent can adapt its behavior (processing control) to its specific environment. According to this system modeling, a dispatcher would be represented by a complex agent, that is, a manager with a dispatching role characterized by decisions that trigger processing of events and control actions on the vehicles.

This paper explains how our management framework can be extended to support dynamic vehicle dispatching. In Section 2, the dynamic dispatching function is analyzed. A network management system in the transportation domain with the integration of a dynamic vehicle dispatcher is presented in Section 3. Section 4 then describes the components of the dispatcher's architecture, as a part of our network management framework. Examples of scenarios illustrating network management agents that collaborate with the dispatcher are presented in Section 5. Concluding remarks are made in Section 6.

2. Dynamic Vehicle Dispatching

2.1 Characteristics of dynamic vehicle dispatching

Vehicle dispatching basically consists in the real-time assignment of new service requests to vehicles. Each dispatching request may imply either a single pick-up location, or both a pick-up and a delivery location, with a time window for service at each location. Most research in this area has focused on the static version of the problem where all requests are assumed to be known in advance. The problem is then to assign and sequence the requests to achieve the desired performance level (which is often stated in terms of operational costs and quality of service). Recent work, however, has addressed the dynamic problem where new requests occur continuously over time and unexpected events are likely to occur. Overviews of dynamic vehicle routing and dispatching problems may be found in [3], [4], [5]. In these problems, it is important to take into account modifications to the decision parameters, which may be either deterministic or non deterministic. An example of the former case is found when the travel times of the vehicles correspond to some (a priori determined) function of the time of the day [6]. In the latter case, the decision parameters change in a non deterministic manner. For example, when an accident occurs on the network, the travel times change suddenly, in a totally unpredictable way. Consequently, the dispatcher must take recourse actions which may imply reassigning customers to different vehicles. Another situation is when the dispatcher is presented with a heavy burden of events to be processed. These may come from brand new events (such as the occurrence of an incident on the network) or consequences of previous ones. In this case, the dispatcher must select the events that must be processed first.

2.2 Main requirements of dynamic vehicle dispatching

From the current literature, it appears that the various functions of the dispatcher are rarely considered globally, that is, by taking into account the various kinds of deterministic and non deterministic events that can occur, as well as their interactions and implications. Clearly, a dispatcher does not mechanically assign requests to vehicles, but does so by considering various decision parameters (e.g., time of the day, presence or absence of congestion close to the request location, etc.). A dispatcher must also be able to perform various recourse actions, when needed (e.g., redirect a vehicle when a customer cancels a request, reassign the requests found on the planned route of a vehicle when that vehicle broke down, etc.). All this must be taken into account for a good performance of the overall network management system, which includes the dispatching function.

Thus, the design and development of a separate dispatching software, outside of any transportation network management and control environment, have little chance of adequately modeling the reality. Clearly, the design of a dispatcher involves many complex issues. It must satisfy real-time requirements, since the events need to be processed quickly. It requires specialized techniques such as operations research (OR) algorithms for scheduling vehicles, calculating shortest paths, etc., as well as inferential techniques, like production rules, to evaluate the importance of new events and trigger the appropriate actions. Finally, it must show abilities to adapt to the current situation and still, under time constraints.

In the next sections, we explain how a dynamic vehicle dispatching function can be integrated within a transportation network management system to meet these requirements. We also explain how a previously developed framework for designing management systems can be easily extended to meet these requirements, thus avoiding the need to redesign the system from scratch.

3. Transportation Network Management

A management system supports control and monitoring functions that are applied to a managed environment. Thus, it requires the collection and assimilation of a large amount of data and expertise. A sophisticated management system helps operators to manage large networks more efficiently, in order to guarantee service quality to the network users. High level management functions include resource optimization, planning, maintenance, etc., and is aimed at avoiding any performance degradation.

A network management system is made up of a set of functions, where each function is a basic unit for providing the required management services. Appropriate functions should be precisely defined for a given network management system. Examples may be found in different application domains such as the electricity [7] and telecommunications [8] industries. Network management systems integrate many different functions, each requiring different techniques and methodologies related to pattern recognition, optimization, logic, etc. Examples of such functions include:

- information gathering and analysis,
- incident detection,
- traffic diversion,
- route guidance and planning (c.f., shortest path calculations),
- dispatching of commercial and emergency vehicles.

Emerging technologies in telecommunications, like SONET (Synchronous Optical Network) and ATM (Asynchronous Transfer Mode), now provide high-speed communications which are necessary for sophisticated network management systems, such as intelligent transportation systems (ITSs). The use of such systems helps, in particular, to reduce traffic congestion through optimized traffic management.

In networks controlled by ITSs, data are carried from roadside equipment such as speed sensors, traffic counters, video cameras, to a traffic operation center where these data are processed and responses are generated. Typically, the communications backbone is self-healing in the face of failures, which is essential for perceiving all information that can affect a new decision or a decision that has already been taken. An example of such an equipped system is presented in Figure 1, where a hub represents network equipment. Links between the equipment and the processing center are typically made of fiber optic. The functions of such a sophisticated network management system must be realized under stringent time constraints. Thus, it is important to find a good trade-off between the processing resources used and the expected quality of the response. For example, if the processing time is too long, the response may not be applicable anymore, because the situation has changed too much. Conversely, if the processing time is too short, the response may be of poor quality. Here, we examine the issue of integrating a dynamic vehicle dispatcher within such an ITS to satisfy the requirements presented in Section 2.2. Clearly, a transportation management network with an ITS-like architecture is required to provide the dispatcher with the required information. Furthermore, such a network management system can be characterized as a distributed real-time system.

The design of real-time systems that integrate different techniques is known as a cost effective and complex task [9]. In addition to the real time requirements, the entities are distributed in a large scale system, thus making the design even more complex. An important research topic in software engineering is reusability [10]. Reusable frameworks avoid the design and development of each individual system within a family of systems that share common criteria [11], [12]. In the next section, we present our framework for the development of complex management applications. We will then see how this framework can be used to support dynamic vehicle dispatching.

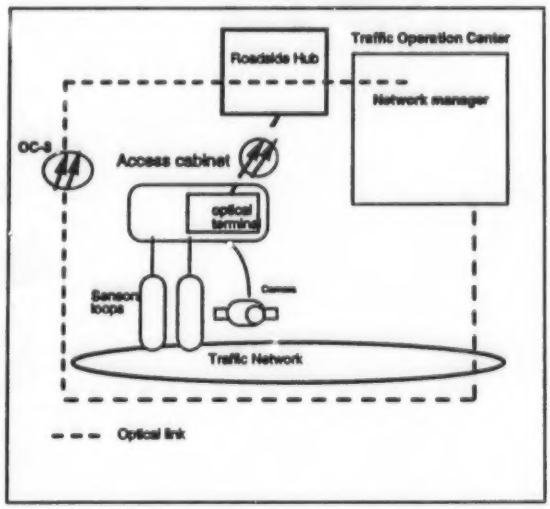

Figure 1. A model of transportation network management

4. An Architecture for Dynamic Vehicle Dispatching

We now use the framework defined in [1], [2] to model a dispatcher working in a transportation network management environment.

4.1 The framework

Our framework is application-independent and designed to satisfy the main requirements of complex applications. This framework models a management system operating under temporal constraints. It is developed in Java RMI (Remote Method Invocation), an object-oriented language, to enable future extensions.

We highlight the abilities integrated within this framework to satisfy the previous requirements. In particular, the scheduling and "event filtering" techniques to process events before their deadline and the adaptive behavior required to optimize the system processing

resources and the overall performance. Dynamic vehicle dispatching clearly requires such abilities and would thus greatly benefit from this application-independent management system. In Figure 2, we present an overview of the main components of our framework called agents.

An *environment agent* simulates the behavior of the environment components which are the managed entities. A *complex agent* is a system component characterized by its ability to process events notified by the environment agents. A complex agent is defined by its role, its knowledge base and its communication scheme with other agents. A *supervisor agent* is a system component responsible for event delegations. When a complex agent receives events to be processed, and its state indicates its inability to process the events agent will assign them to other appropriate complex agents with the same role. For the time being, complex agents cannot communicate directly among themselves, to avoid performance degradation.

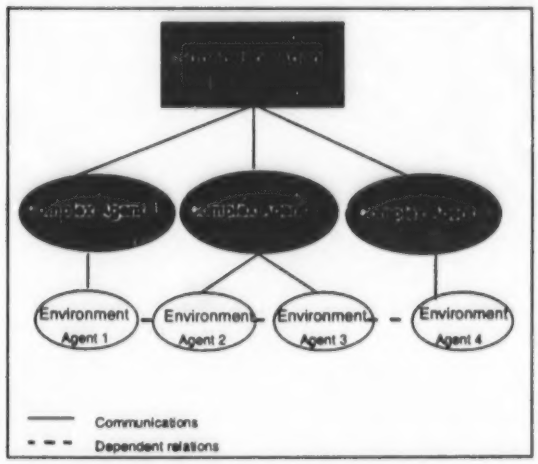

Figure 2. Architecture for a complex application framework [Benyahia & Hilali 2000]

An important design decision consists in the definition of significant processing components, namely complex agents. The latter should represent in a realistic manner the real-world components that they model.

The architecture of a complex agent supports the following functions:

- perceive significant events,
- evaluate events in a global context,
- adapt the processing strategies to a changing environment to meet the deadlines.

The main components of a complex agent architecture are called tasks (see Figure 3). Tasks are made up of several sub-components called actions. A scenario for the operations of a complex agent, as illustrated in Figure 3, is the following. When an environment event occurs, it is first processed by the *receiver* which decides whether to accept or to refuse it, according to the action that filters new events. The receiver has a queue of events to enable asynchronous communication. When an event is accepted, it is transmitted to the second component, the evaluator, which has two roles: environment evaluation and processing evaluation. The *environment evaluator* evaluates events in the context defined by the system and

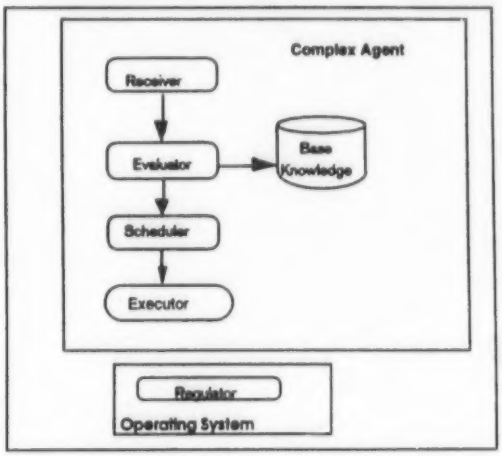

Figure 3. Architecture of a complex agent

its environment. This evaluator assigns dynamic parameter values to each event, such as their importance, priority, risk associated with non-processing and deadline. Then, the *processing evaluator* is invoked to define the specific actions to be executed for processing one event among potential candidates. Given the event deadline, a complex agent must be able to define the processing order for that event in the queue of already received events. The architecture component which is responsible for ordering events is the *scheduler* which uses the dynamic parameter values provided by the environment evaluator. Finally, each event is processed by the *executor* which triggers the actions defined by the processing evaluator. The actions associated with each event can be stored in a library shared by all agents or it may be a technique specific to one agent according to its role in the network management. For more details on complex agents, see [2].

In the context of a dynamic vehicle dispatching environment, agents can represent the (monitored) vehicle traffic, routes, vehicles, etc. For each complex agent specialized in vehicle dispatching, we associate a set of vehicles and a geographic area which is dynamic and may change depending on various factors (time of the day, current vehicles routes, etc.). Customer calls are

assumed to be transferred to the closest dispatcher to accelerate service. Given this, the following event processing scheme takes place:

1) Receive an event from an agent that detects significant events, such as the incident detector [13] (e.g., occurrence of congestion, new customer call, etc.).
2) Evaluate accepted events using dynamic parameter values such as importance, deadline, priority. (e.g., "congestion importance = 'high risk' "). This evaluation concerns the vehicles associated with each complex agent and its current dispatching area (i.e., the dynamic subnetwork where the vehicles travel).
3) Evaluate the list of actions to execute according to the dynamic parameter values (e.g., "rerouting of the vehicles that will enter into the congested area").
4) Schedule new events with already received ones (e.g., "insert the defined action to be executed immediately").
5) Execute suitable actions (e.g., "send control actions to appropriate vehicles").

4.2 Application of the framework to vehicle dispatching

Clearly, the dispatcher is a complex agent. Thus, we must first recognize the different functions of the dispatcher to evaluate its processing requirements. These functions may be categorized as follows:

i) Traffic analysis. The dispatcher must be continuously aware of current traffic conditions.

ii) Forecasting. The dispatcher should be able to forecast future incoming requests to take better decisions at the current time (e.g., a vehicle will be asked to stay idle at its current location, even if other requests are pending, because a new request in the same area is expected soon).

iii) Assignment of new requests. This is the main function of the dispatcher and consists in assigning new requests to vehicle routes, as they occur.

iv) Route planning. The incremental insertion of new customer requests into the planned routes may lead to grossly suboptimal solutions. The dispatcher should thus be able to reconsider the situation and take actions that will improve the performance level. For example, a reordering of the requests on the planned routes may be considered from time to time to improve either the operational costs or the quality of service (or both, if possible). Also, a dispatcher can inform its supervisor of important changes. For example if there are opportunities to assign new tasks to its vehicles. This information is then forwarded to other complex agents (dispatchers), allowing them to review their current planning and possibly improve it.

v) Recourse. In various situations, the dispatcher must be able to react quickly by performing fast and simple recourse actions. For example, assume that a vehicle is heading towards its next destination and a new request just occurs in the vicinity of its current location. Then, the dispatcher should be able to tell the driver to take the new request "en-route" to its destination. Recourse actions should also be taken when facing unexpected events (vehicle breakdown, incident on the transportation network,...).

Several other agents in the management framework may be invoked to provide the required information to the dispatcher. In order to trigger suitable techniques to respond to events, the dispatcher has access to a library of technical components that can be shared with other agents of the management system. For each network management function realized by the dispatcher, there are several possible techniques that can be chosen depending on the current context, which is defined by the state of the dispatcher and its environment. A dynamic vehicle dispatching function can thus be derived from our framework, by integrating the technical components specific to this application, like those shown in Table 1 (for more details on these techniques, see [14]).

Network management functions	Technical component
Traffic analysis	• Pattern recognition • Statistics (e.g. data mining) • Artificial intelligence
Forecasting	• Time series • Neural networks
Assignment of new requests	• Shortest path algorithms • Insertion methods
Route planning	• Decomposition • Exact methods (e.g., branch-and-bound, dynamic programming) • Approximate methods (e.g., metaheuristics: tabu search, genetic algorithms, ...[16])
Recourse	• Traffic diversion • Local optimization

Table 1. Technical components

In addition to the technical components, a rule base must be defined to evaluate the dynamic parameters associated with the specified events and to indicate which technical components should be triggered to respond to these events. This rule base is used by the processing evaluator. Table 2 presents some examples of sequences

of actions which may be associated with unexpected events received by the dispatcher.

Unexpected event	Sequence of actions
Accident on the network	• Check impact of congestion on planned routes • Trigger reoptimization on affected routes, if any
Service delay at customer location	• Communicate with incident detector agent • Trigger actions to estimate duration • Trigger suitable techniques such as traffic diversion • Communicate with customer to inform him and take corrective actions
Vehicle breakdown	• Reassign requests of broken vehicle
Customer notification	• Quick reoptimization of the planned route

Table 2. Association event-action

5. Operating Scenarios

The dispatcher can operate either in a stable or a perturbed regime. A stable regime means that the situation evolves as planned. An example may be found in [15] where the events are limited to the occurrence of new service requests. In this work, dynamic parameter values characterizing each vehicle are calculated and stored in a data base which is then used for decision making. Conversely, a perturbed regime is characterized by the occurrence of unexpected events such as those presented in Table 2.

The quality of the dispatching function depends on the behavior of the dispatcher and the way it triggers the required technical components, as defined by its processing evaluator. When it operates in a stable regime, it will typically always trigger the same basic operations. However, when it operates in a perturbed regime, it needs to be reactive to adapt its processing to changes in the environment. A typical example is the occurrence of an accident which may lead to severe congestion. As the time needed for the operations of incident detection and route clearance gets longer, the risk of propagation of the congestion gets higher. Thus a distributed processing scheme among several agents of the considered ITS is required. It means that the dispatcher will delegate some actions to other agents that will process them concurrently in order to minimize the response time.

The design of the dynamic dispatcher presented here may be based on different operating scenarios representing different combinations of processing by the dispatcher and other agents that collaborate with it. An important parameter that must be taken into account for selecting the appropriate scenario is the capacity of the dispatcher to process important events before their deadline. Let's consider the situation defined as the occurrence of the event "service delay at a customer location". We present three alternative ways of processing this event.

In the first scenario presented in Figure 4, the dispatcher executes all the defined actions (i.e., the technical components are applied in sequence). However, if another event is detected, like the occurrence of a new request between the location being currently serviced by a given vehicle and the next one, the dispatcher may miss the opportunity to redirect the vehicle towards the new request, if its processing cannot be interrupted. With a sequential processing behavior, there is also a risk of important delays in the processing of urgent incidents on the network.

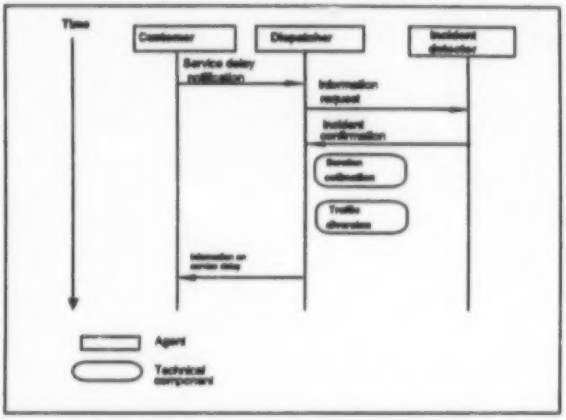

Figure 4. Scenario 1: Sequential actions

In the second scenario presented in Figure 5, the dispatcher invokes a method from another agent called "duration evaluator" to estimate the duration of the current incident. While this agent is executing its role, the dispatcher is ready to receive and process other events. When the duration evaluator has a result, it is transmitted to the dispatcher which pursues its processing according to the actions presented in Table 2. If no event occurs during the dispatcher's processing, the system can be considered as efficient since there is no need (in this case) to call another agent and waste time and communication resources. Note, however, that performance degradation may be observed if new events that need fast reactive processing occur while the

dispatcher is busy (i.e., at the end of the dispatcher's processing, it may be too late to delegate urgent actions to other complex agents).

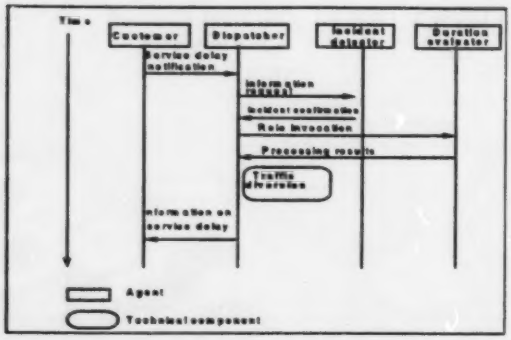

Figure 5. Scenario 2: Collaboration of one agent with the dispatcher

Finally, in the third scenario presented in Figure 6, we consider an additional agent that realizes traffic diversion. This agent collects data on the affected routes in order to perform diversion. Depending on the location, it may need to communicate with other complex agents to get information from their data and knowledge base. Thus, the communication costs can be higher than those required by a single dispatcher that receives and processes events.

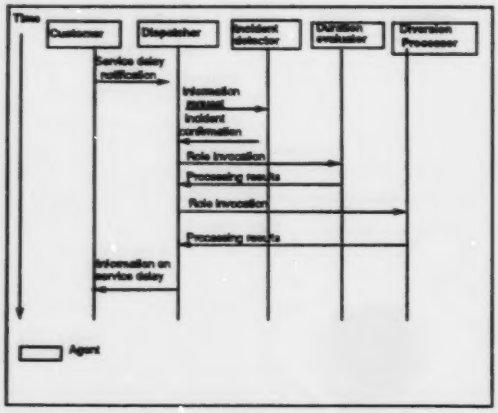

Figure 6. Scenario 3: Collaboration of two agents with the dispatcher

In order to take a decision about the appropriate dispatcher's behavior when it operates in a perturbed regime, it is important to perform experiments and to analyze the results. To this end, developing a library of plausible operating scenarios to be experimented represents an important step towards the development of a dynamic vehicle dispatcher.

6. Conclusion

The dynamic vehicle dispatching function requires large amount of information to be processed. This function is hard to design because it is based on decision parameters that may change dynamically and must be processed under time pressure. An overall view of this function, with its interaction with other network management functions, must be developed to guarantee the required operating performance.

Developing applications for complex real-world problems is a difficult task, since many different problem solving techniques must be integrated. In this paper, we have considered a distributed framework designed to support this task. This framework is reusable and based on object-oriented technology. An extension to this framework which integrates components specific to a dynamic vehicle dispatcher is presented. A library of technical components is also introduced with their association to unexpected events.

Finally, examples of operating scenarios are presented and analyzed. Simulation studies based on such potential scenarios will be performed to analyze the behavior of a dynamic dispatcher when it operates in a perturbed regime where unexpected events occur.

7. References

[1] I. Benyahia, M. Hilali and E. Taillard, " Using adaptive processing to optimize the performance of distributed systems", presented at the Optimization Days, Montréal, Canada, 2000.

[2] I. Benyahia, M. Hilali, "An adaptive framework for distributed complex applications development", IEEE TOOLS USA, pp 339-349, 2000.

[3] M. Gendreau, J Y Potvin , "Dynamic vehicle routing and dispatching", in Fleet Management and Logistics, T.G. Crainic and G. Laporte eds., Kluwer, pp. 115-126, 1998.

[4] W. B. Powell, P. Jaillet, and A. Odoni, "Stochastic and dynamic networks and routing", in Network Routing, M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser eds., North-Holland, pp. 141-295, 1995.

[5] A. Weintraub, A. Aboud, C. Fernandez, G. Laporte and E. Ramirez, "An Emergency Vehicle Dispatching System for an Electric utility in Chilel", Journal of Operational Research Society, N. 5, 1999.

[6] C. Malandraki and M.S. Daskin, "Time-dependent vehicle routing problems: formulations, properties and heuristic algorithms", Transportation Science 26, pp.185-200, 1992.

[7] F. R. Belch and A. Crewe, "Network information management systems", Computing & Control engineering Journal, 1993.

[8] N. Modiri, "The ISO reference model entities", IEEE Network Magazine, 1991.

[9] J. Stankovic, "A serious Problem For Next Generation Systems", IEEE COMPUTER, pp 10-19, 1988.

[10] B. Meyer "Reusable software: the base object-oriented component libraries", Prentice-Hall Object-Oriented Series, 1994.

[11] K. Choy, M. Halkyard, M. Krishnamoorthy, P. Shankar, "Reusable Management Frameworks For Third-Generation Wireless Networks", Bell Labs Technical Journal, V. 4, No 4, pp 171-189, 1999.

[12] H. Gomaa and G. A. Farrukh, "Methods and Tools For the Automated Configuration of Distributed Applications From Reusable Architectures and Components", IEE Proceedings-Software, V. 146, No 6, pp 277-289, 1999.

[13] K. Ozbay and P. Kashroo, "Incident management in intelligent transportation systems", Artech House, 1999.

[14] R. Séguin, J.Y. Potvin, M. Gendreau, T.G. Crainic, and P. Marcotte, "Real-time decision problems: an operational research perspective", Journal of the Operational Research Society 48, pp 162-174, 1997.

[15] I. Benyahia, J.Y. Potvin (1998), "Decision support for vehicle dispatching using genetic programming", IEEE Transaction on Systems, Man and Cybernetics 28, pp. 306-314, 1998.

[16] G. Laporte and I.H. Osman (eds.), "Metaheuristics in Combinatorial Optimization", Annals of Operations Research 63, 1996.